

APPLICATION  
FOR  
UNITED STATES LETTERS PATENT

APPLICANT NAME: Kent F. Hayes, Jr.

TITLE: TUNABLE ENGINE, METHOD AND PROGRAM  
PRODUCT FOR RESOLVING PREREQUISITES FOR  
CLIENT DEVICES IN AN OPEN SERVICE GATEWAY  
INITIATIVE (OSGi) FRAMEWORK

DOCKET NO.: RSW920030234US1

INTERNATIONAL BUSINESS MACHINES CORPORATION

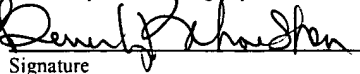
**CERTIFICATE OF MAILING UNDER 37 CFR 1.10**

I hereby certify that, on the date shown below, this correspondence is being deposited with the United States Postal Service in an envelope addressed to Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 as "Express Mail Post Office to Addressee" Mailing Label No. EV393299588US

on March 22, 2004

Beverly Kehoe Shea

Name of person mailing paper



Signature

03/22/2004

Date

**TUNABLE ENGINE, METHOD AND PROGRAM PRODUCT FOR RESOLVING  
PREREQUISITES FOR CLIENT DEVICES IN AN OPEN SERVICE GATEWAY  
INITIATIVE (OSGi) FRAMEWORK**

**Background of the Invention**

**1. Field of the Invention**

[0001] In general, the present invention relates to a tunable engine, method and program product for resolving prerequisites for client devices utilizing an OSGi runtime/framework. Specifically, the present invention allows the prerequisite resolution process to be uniquely configurable for different client devices.

**2. Related Art**

[0002] As computer networking has become more advanced, a standard known as the Open Service Gateway Initiative (OSGi) has been developed. The OSGi is an industry plan to provide a standard way to deliver managed services to devices and local networks. With such a standard, home users could, for example, change the setting on their thermostat from a remote location (e.g., the workplace). In general, the OSGi provides a good framework for developing application components. Under the OSGi, a basic component is known as an OSGi bundle. An OSGi application can be made up of combinations/suites of bundles that might share common functionality. To this extent, the OSGi allows developers to define the dependencies between the bundles such as the packages and services required by the bundles. In a typical implementation, an OSGi

architecture will include, among other components, a server and one or more client devices. Each client device will have an OSGi environment within which OSGi applications are deployed. Using a management program on the sever, the functions of the OSGi applications can be controlled.

[0003] Unfortunately, as convenient as the OSGi framework can be, it currently fails to provide a tunable way to resolve prerequisites for OSGi bundles that are to be loaded and run on a client device. For example, for an OSGi bundle to properly run on a client device, certain packages and/or services might be necessary and levels of resources such as RAM, sockets, threads, disk space, etc.. If the client device is lacking any of the necessary prerequisites, other OSGi bundles (or combinations thereof) that provide the missing prerequisites should be utilized.

[0004] However, it may be the case that there is more than one combination of OSGi bundles that could provide the missing prerequisites. These different combinations of bundles might consume different levels of resources (i.e., RAM, sockets, threads, disk space, etc.). The collection of bundles chosen to supply the missing prerequisites should make the best use of the resources available without exceeding the resource levels available on the device. Some resources available on the device might be relatively more sparse and/or valuable then other resources on the device. Thus, there exists a need not only to resolve and supply the missing prerequisites, but to do so in a manner consistent with the aforementioned characteristics (i.e., level and relative value of a type of resource). As such, the server should reconcile the capabilities of the client device with the necessary prerequisites of the OSGi bundles. This is not a simple goal to realize

especially since each different client device could have different capabilities or available computer resources. Moreover, an OSGi bundle identified to address a prerequisite that the client device lacks could itself have certain prerequisites that the client device lacks. Accordingly, not only should the prerequisite resolution process be configurable for each type of client device (or individual client device as appropriate), but it should also be done recursively to ensure that all prerequisites are resolved.

[0005] In view of the foregoing, there exists a need for a tunable engine, method and program product for resolving prerequisites for client devices utilizing an Open Service Gateway Initiative (OSGi) framework/runtime. Specifically a need exists for a system that can recursively resolve prerequisites based on the individual needs and resource characteristics of the target client device. To this extent, a further need exists for the system to accommodate any quantity of client devices.

### **Summary of the Invention**

[0006] In general, the present invention provides a tunable engine, method and program product for resolving prerequisites for client devices in an Open Service Gateway Initiative (OSGi) framework. Specifically, under the present invention, each type of client device (or each individual client device) will have a corresponding configuration (e.g., a Java properties file in one possible embodiment of the invention) that is accessible to the server. Each configuration sets forth various types of computer resources of the device(s), as well as corresponding normalization and importance factors for the associated client device (or group of client devices). If certain OSGi bundle(s) is/are to

be loaded and run on a particular client device, various prerequisites (e.g., services packages, etc.) will likely be necessary. Any prerequisites that the client device lacks will be obtained using potential combinations of other OSGi bundles registered with the server. Under the present invention, the most appropriate combination of OSGi bundles will be identified to supply the missing prerequisites by processing the configuration and determining, based on the amount of computer resources (of the client device) that will be consumed by each of the possible combinations, an appropriate combination. Specifically, the combination of OSGi bundles that best meets the needs of the client device as set forth in the corresponding configuration will be selected.

[0007] A first aspect of the present invention provides a tunable prerequisite resolution engine for client devices in an Open Service Gateway Initiative (OSGi) framework, comprising: a prerequisite computation system for determining, on a server, necessary prerequisites for one or more OSGi bundle(s) to be run on a client device; a bundle identification system for identifying possible combinations of OSGi bundles on the server that fulfill any of the necessary prerequisites that are lacking from the client device; a data access system for accessing a configuration corresponding to the client device, wherein the configuration sets forth computer resource types, and corresponding normalization factors and importance factors for the client device (or group of client devices); and a bundle selection system for selecting the best combination of OSGi bundles from the possible combinations of OSGi bundles by processing the configuration and computing based on amounts of the computer resources needed by each of the possible combinations of OSGi bundles.

[0008] A second aspect of the present invention provides a computer-implemented method for resolving prerequisites for client devices in an Open Service Gateway Initiative (OSGi) framework, comprising: determining, on a server, necessary prerequisites for an OSGi bundle to be run on a client device; identifying possible combinations of OSGi bundles on the server that fulfill any of the necessary prerequisites that are lacking from the client device; accessing, from the server, a configuration corresponding to the client device, wherein the configuration sets forth computer resource types and corresponding normalization factors and importance factors for the client device; and selecting a final combination of OSGi bundles from the possible combinations of OSGi bundles by processing the configuration based on amounts the computer resources needed by each of the possible combinations of OSGi bundles.

[0009] A third aspect of the present invention provides a program product stored on a recordable medium for automatically resolving prerequisites for client devices in an Open Service Gateway Initiative (OSGi) framework, which when executed, comprises: program code for determining, on a server, necessary prerequisites for an OSGi bundle to be run on a client device; program code for determining possible combinations of OSGi bundles on the server that fulfill any of the necessary prerequisites that are lacking from the client device; program code for accessing a configuration corresponding to the client device, wherein the configuration sets forth computer resource types and corresponding normalization factors and importance factors for the client device; and program code for selecting a final combination of OSGi bundles from the possible combinations of OSGi

bundles by processing the configuration based on amounts of the computer resources needed by each of the possible combinations of OSGi bundles.

[0010] Therefore, the present invention provides a tunable engine, method and program product for resolving prerequisites for client devices utilizing an Open Service Gateway Initiative (OSGi) framework/runtime.

### **Brief Description of the Drawings**

[0011] These and other features of this invention will be more readily understood from the following detailed description of the various aspects of the invention taken in conjunction with the accompanying drawings in which:

[0012] Fig. 1 depicts an illustrative system for resolving prerequisites for client devices utilizing an OSGi framework/runtime according to the present invention.

[0013] Fig. 2 depicts the system of Fig. 1 in greater detail.

[0014] Fig. 3 depicts a method flow diagram according to the present invention.

[0015] It is noted that the drawings of the invention are not necessarily to scale. The drawings are merely schematic representations, not intended to portray specific parameters of the invention. The drawings are intended to depict only typical embodiments of the invention, and therefore should not be considered as limiting the scope of the invention. In the drawings, like numbering represents like elements.

## **Detailed Description of the Drawings**

[0016] For convenience purposes, the Detailed Description of the Drawings will have the following sections:

I. General Description

II. Detailed Example

### **I. General Description**

[0017] As indicated above, the present invention provides a tunable engine, method and program product for resolving prerequisites for client devices utilizing an Open Service Gateway Initiative (OSGi) framework/runtime. Specifically, under the present invention, each type of client device (or each individual client device) will have a corresponding configuration (e.g., a Java properties file) that is accessible to the server. Each configuration sets forth various types of computer resources of the device(s), as well as corresponding normalization and importance factors for the associated client device. If certain OSGi bundle(s) is/are to be loaded and run on a particular client device, various prerequisites (e.g., services packages, etc.) will likely be necessary. Any prerequisites that the client device lacks will be obtained using potential combinations of other OSGi bundles registered with the server. Under the present invention, the most appropriate combination of OSGi bundles will be identified to supply the missing prerequisites by processing the configuration and determining, based on the amount of computer resources (of the client device) that will be consumed by each of the possible combinations, an appropriate combination. Specifically, the combination of OSGi bundles that best meets



the needs of the client device as set forth in the corresponding configuration will be selected.

[0018] As used herein, the term “prerequisites” refer to packages, services or the like that are needed by a client device in order to properly run an OSGi bundle. For example, OSGi bundle “X” might require service “A” and package “C” to be properly run. As known, under the OSGi, a “package” is similar to a JAVA package and a “service” is a certain type of interface. In addition, the term “computer resources” is intended to refer to resources of the client device such as memory, RAM, and the like.

[0019] Referring now to Fig. 1, an illustrative system 10 for resolving prerequisites for client devices 14A-N in an OSGi framework/runtime according to the present invention is shown. As depicted, system 10 includes server 12 and client devices 14A-N. It should be understood that the architecture shown herein is illustrative only and will likely include other known components not shown. For example, a typical OSGi framework would likely include a device, OSGi agents (on client devices 14A-N), a device management server and one or more application servers. Moreover, it should be understood that a typical OSGi framework could include multiple servers 12 and a network dispatcher. In any event, client devices 14A-N are intended to represent any type of computerized device capable of communicating over a network. For example, client devices 14A-N could be desktop computers (e.g., WIN-32-based, Linux based, etc.), hand held devices, set top boxes, home appliances, security systems, etc. In any event, server 12 and client devices 14A-N typically communicate over any type of network such as the Internet, a local area network (LAN), a wide area network (WAN), a virtual private network (VPN),

etc. As such, communication between server 12 and client devices 14A-N could occur via a direct hardwired connection (e.g., serial port), or via an addressable connection that may utilize any combination of wireline and/or wireless transmission methods.

Moreover, conventional network connectivity, such as Token Ring, Ethernet, WiFi or other conventional communications standards could be used. Still yet, connectivity could be provided by conventional TCP/IP sockets-based protocol. In this instance, client devices 14A-N could utilize an Internet service provider to establish connectivity to server 12.

[0020] Under the present invention, each client device 14A-N (or type of client device, or group of client devices) will have a corresponding configuration 24A-N (e.g., a properties file) that is accessible to server 12. Each configuration 24A-N sets forth certain computer resource types of the associated client device as well as corresponding normalization and importance factors. For example, for client device 14A, configuration 24A could resemble the following:

<b><u>Computer Resource Type</u></b>	<b><u>Normalization</u></b>	<b><u>Importance</u></b>
BundleStorage	0.0001	0
NewSpaceSize	0.0001	10
OldSpaceSize	0.0001	0
OpenFiles	10	0
Quota	0.0001	0
RAM	0.0001	0
Sockets	10	0
Threads	10	5
Total RAM	0.0001	10

The normalization factors allows each of the computer resources to be compared on an equal basis. For example, according to the above factors, 10 sockets is the roughly

equivalent of 0.0001 K of RAM. The importance factors allow an importance level to be assigned to each computer resource with respect to the particular client device 14A-N. For example, using the above configuration, it can be seen that conserving NewSpaceSize and Total Ram are the most important factors to client device 14A. Conserving the number of threads used is of lesser importance, while the remaining computer resources (having been assigned an importance factor of 0) have no importance to client device 14A one way or the other. As will be further described below, configurations 24A-N such as the above allows the prerequisite resolution process to be individually tuned for each client device 14A-N. The process can additionally be tuned for each type of client device, arbitrary collections of client devices, etc.

[0021] In any event, assume that server 12 is attempting to load OSGi bundle 18 on client device 14A. As known, an OSGi bundle is essentially a .JAR file with certain characteristics/functionality which enable it to effectively interact with and be controlled by the OSGi framework. Server 12 includes (tunable) resolution engine 16, which can incorporate parts of, or be included within any type of OSGi management program now known or later developed. It is known that in order for OSGi bundle 18 to be properly run on client device 14A, certain prerequisites might be needed. Under the present invention, resolution engine 16 will determine and resolve such prerequisites. Typically, the prerequisites for OSGi bundle 18 will first be determined by referencing information in a repository or the like. However, in another embodiment, the prerequisites could be determined by analyzing OSGi bundle 18.

[0022] Regardless, once the prerequisites are known, resolution engine 16 can communicate the same to client device 14A. This communication could be made in response to a request by client device 14 (e.g., by an agent thereon). Upon receipt, client resolution system 22A will analyze the prerequisites and determine whether it has the needed prerequisites. Specifically, client resolution system 22A will determine whether any prerequisites are lacking (from client device 14A). In a typical embodiment, this information is determined from cache on client device 14A. To this extent, client device 14A should maintain up to date information concerning its available computer resources, packages and services. Once the analysis is complete, client resolution system 22A will generate and send a response back to server 12 that indicates the missing prerequisites. Resolution engine 16 will receive and cache the information derived from the response, and then attempt resolve the prerequisites.

[0023] In a typical embodiment, resolution engine 16 will resolve the prerequisites by first identifying one or more possible combinations of OSGi bundles (e.g., 18 and 20) that can provide the missing prerequisites. Thereafter, resolution engine 16 will select the most appropriate combination of OSGi bundles from the possible combinations based on the computer resource “needs” of client device 14A. Specifically, resolution engine 16 will process configuration 24A for client device 14A based on the amounts of computer resources of client device 14A each of the possible combinations will need/consume. The possible combination that best fits the needs of client device 14A will then be selected as the best combination.

[0024] It should be noted that the collection of one or more bundles providing the needed prerequisites may have prerequisites of their own. If it is determined, either from the cached information, or by asking client device 14A (specifically client resolution system 22A on the client device 14A) that there are more missing prerequisites that must be fulfilled, resolution engine 16 will again find combinations of bundles following the aforementioned process for selecting the most appropriate combination of bundles providing the missing prerequisites. This process continues recursively until all prerequisites have been resolved.

[0025] The final combination of OSGi bundles could include any combination of OSGi bundle 18 and other OSGi bundles 20. For example, if client device 14A is lacking a certain service needed to properly run OSGi bundle 18, resolution engine 16 will identify another OSGi bundle 20 that has the necessary service and than can operate within the computer resource limitations/needs of client device 14A. Similarly, if OSGi bundle 18 would require more RAM than client device 14A wishes to have consumed, resolution engine 16 could attempt to locate one or more other OSGi bundles 20 that could be substituted for OSGi bundle 18 and operate within the resource limitations of client device 14A. To this extent, the final combination of OSGi bundles could include OSGi bundle 18 in addition to one or more other OSGi bundles 20, or one or more other OSGi bundles 20 without OSGi bundle 18.

[0026] In a typical embodiment, the resolution process occurs recursively meaning that the resolution is continuously repeated until all prerequisites are resolved, or it is determined that the prerequisites cannot be resolved within the resource limitations of the

device. Recursive resolution is especially useful since any quantity or hierarchy of prerequisites might need resolution (e.g., other OSGi bundles 20 could themselves have prerequisites). Moreover, in a typical embodiment, once server 12 asks client device 14A about a particular resource or prerequisite, server 12 will not ask about that prerequisite or resource again (e.g., due to caching on server 12). In any event, once the prerequisites are completely resolved, resolution engine 16 will load the final set of OSGi bundles on client device 14A. Typically, the loading process includes server 12 sending client device 14A an instruction(s) pertaining the order in which OSGi bundles 18 and 20 should be loaded. However, if the prerequisites cannot be completely resolved, none of OSGi bundles 18 or 20 would be loaded.

## **II. Detailed Example**

[0027] Referring now to Fig. 2, a more detailed diagram of Fig. 1 is shown. As shown, server 12 generally comprises central processing unit (CPU) 30, memory 32, bus 34, input/output (I/O) interfaces 36, external devices/resources 38 and storage unit 40. CPU 30 may comprise a single processing unit, or be distributed across one or more processing units in one or more locations, e.g., on a client and computer system. Memory 32 may comprise any known type of data storage and/or transmission media, including magnetic media, optical media, random access memory (RAM), read-only memory (ROM), a data cache, etc. Moreover, similar to CPU 30, memory 32 may reside at a single physical location, comprising one or more types of data storage, or be distributed across a plurality of physical systems in various forms.

[0028] I/O interfaces 36 may comprise any system for exchanging information to/from an external source. External devices/resources 38 may comprise any known type of external device, including speakers, a CRT, LCD screen, handheld device, keyboard, mouse, voice recognition system, speech output system, printer, monitor/display, facsimile, pager, etc. Bus 34 provides a communication link between each of the components in server 12 and likewise may comprise any known type of transmission link, including electrical, optical, wireless, etc.

[0029] Storage unit 40 can be any system (e.g., database, repository, etc.) capable of providing storage for information under the present invention. Such information could include, for example, additional OSGi bundles 20, configurations 24A-N (e.g., properties files), prerequisite information, etc. As such, storage unit 40 could include one or more storage devices, such as a magnetic disk drive or an optical disk drive. In another embodiment, storage unit 40 includes data distributed across, for example, a local area network (LAN), wide area network (WAN) or a storage area network (SAN) (not shown). Although not shown, additional components, such as cache memory, communication systems, system software, etc., may be incorporated into server. In addition, it should also be appreciated that although not shown, client device 14 would likely include computerized components similar to server 12.

[0030] Shown in memory 32 of server 12 is (tunable) resolution engine 16. As indicated above, resolution engine 16 can be part of, or incorporate parts of any OSGi management program now known or later developed. It should be understood that each of the systems within resolution engine 16 includes program code/logic for carrying out the functions

described herein. To this extent, the systems could be realized as plugins or the like. In any event, assume in this example that it is desired to load OSGi bundle 18 on client device 14A. In this case, prerequisite computation system 42 will first determine the prerequisites (e.g., services and/or packages) needed for proper operation thereof. As indicated above, the prerequisites can be computed by accessing information in cache memory, storage unit 40 or the like. In another embodiment, prerequisite computation system 42 could include logic to analyze OSGi bundle 18 and compute the prerequisites. [0031] Regardless, once the prerequisites are determined, prerequisite computation system 42 can check a running table of supplied services and packages, as maintained by prerequisite computation system 42 (e.g., in storage unit 40), and remove any prerequisites that are known to exist. Thereafter, communication system 44 will send a list of the needed prerequisites to client device 14A. As indicated above, this communication could be made in response to a request by client device 14 (e.g., by an agent thereon). Upon receipt, analysis system 64A within client resolution system 22A will analyze the prerequisites and determine whether all needed prerequisites are present. To this extent, client device 14A should maintain up to date information concerning its available computer resources, packages and services. After the analysis, response system 66A will generate and send a response back to server 12. The response will identify any resource limitations of client device 14A. For example, if client device 14 lacked a necessary service or package, those limitations would be noted in the response that is sent back to server 12. Similarly, if it is indicated that an amount of RAM, threads, etc., are



needed to run the bundle(s), then the level of the pertinent resource is indicated to the server 12 from client device 14A.

[0032] Upon receipt, caching system 46 will cache the information derived from the response for later use in the prerequisite resolution process (and update the running table).

Thereafter, bundle identification system 48 will identify possible combinations of OSGi bundles that fulfill the missing prerequisites. For example, if client device 14A lacked service “A,” and package “C,” bundle identification system 48 would attempt to find possible combinations of OSGi bundles (18 and 20) that will provide those prerequisites. To this extent, assume that there are three possible combinations of OSGi bundles that will provide all necessary prerequisites, namely, possible combinations “1,” “2” and “3.”

[0033] Once these possible combination have identified, bundle selection system 54 will select as the most appropriate combination of OSGi bundles (i.e., a final combination of OSGi bundles), by processing the configuration corresponding to client device 14A in view of the amounts of computer resources of client device 14A each possible combination will require. In particular, data access system 50 will first access the configuration 24A corresponding to client device 14A. In a typical embodiment, the properties files 24A-N are stored locally to server 12 (e.g., in storage unit 40). However, this need not be the case and the properties files can be accessed remotely. Regardless, as indicated above, each client device 14A-N will have a corresponding configuration 24A-N (i.e., each configuration may be configured for a particular device, all devices of a specific device type, or by any other grouping option). As such, the configuration 24A for client device 14A will likely be accessed from within a group of configurations. To

configure (create, modify or delete) configurations 24A-N, an administrator or the like (not shown) would utilize configuration system 52. To this extent, configuration system 52 could provide a set of interface pages or the like that the administrator would use to set the configurations 24A-N. This is what allows resolution engine 16 to be tunable. Specifically, resolution engine 16 is tuned based on the particular client device 14A-N on which an OSGi bundle is being loaded.

[0034] Listed again below is configuration 24A for client device 14A, which was previously shown in Section I above:

<b><u>Computer Resource</u></b>	<b><u>Normalization</u></b>	<b><u>Importance</u></b>
BundleStorage	0.0001	0
NewSpaceSize	0.0001	10
OldSpaceSize	0.0001	0
OpenFiles	10	0
Quota	0.0001	0
RAM	0.0001	0
Sockets	10	0
Threads	10	5
Total RAM	0.0001	10

This file would be processed by bundle selection system 54 in view of the amounts of computer resources of client device 14A that each of the possible combinations would consume. Specifically, amount detection system 56 would first determine the amounts of each of the computer resources set forth in the configuration each combination would consume. This can be done based on information contained in storage unit 40 corresponding to each combination. For example, assume each of the possible combinations would consume the corresponding amounts of computer resources:

<u>Computer Resource</u>	<u>Combination 1</u>	<u>Combination 2</u>	<u>Combination 3</u>
BundleStorage	3000	4000	3000
NewSpaceSize	3000	2000	4000
OldSpaceSize	4000	5000	5500
OpenFiles	5	4	5
Quota	6	5	6
RAM (K)	640	720	1000
Sockets	4	4	5
Threads	5	7	8
Total RAM (K)	640	720	1000

Once determined, this information would be processed using the normalization and importance factors set forth in configuration 24A. Specifically, product determination system 58 would determine products for each of the possible combinations of OSGi bundles by multiplying each of the amounts set forth above by the corresponding normalization factors and importance factors set forth in configuration 24A.

This would yield the following results:

<u>Resource</u>	<u>Combination 1</u>	<u>Normalization</u>	<u>Importance</u>	<u>Product</u>
BundleStorage	3000	(0.0001)	0	0
NewSpaceSize	3000	(0.0001)	10	3.0
OldSpaceSize	4000	(0.0001)	0	0
OpenFiles	5	(10)	0	0
Quota	6	(0.0001)	0	0
RAM (K)	640	(0.0001)	0	0
Sockets	4	(10)	0	0
Threads	5	(10)	5	250
Total RAM (K)	640	(0.0001)	10	0.64

<u>Resource</u>	<u>Combination 2</u>	<u>Normalization</u>	<u>Importance</u>	<u>Product</u>
BundleStorage	4000	(0.0001)	0	0
NewSpaceSize	2000	(0.0001)	10	2.0
OldSpaceSize	5000	(0.0001)	0	0
OpenFiles	4	(10)	0	0
Quota	5	(0.0001)	0	0
RAM (K)	720	(0.0001)	0	0
Sockets	4	(10)	0	0

Threads	7	(10)	5	350
Total RAM (K)	720	(0.0001)	10	0.72

<u>Resource</u>	<u>Combination 3</u>	<u>Normalization</u>	<u>Importance</u>	<u>Product</u>
BundleStorage	3000	(0.0001)	0	0
NewSpaceSize	4000	(0.0001)	10	4.0
OldSpaceSize	5500	(0.0001)	0	0
OpenFiles	5	(10)	0	0
Quota	6	(0.0001)	0	0
RAM (K)	1000	(0.0001)	0	0
Sockets	5	(10)	0	0
Threads	8	(10)	5	400
Total RAM (K)	1000	(0.0001)	10	1.0

[0035] After the products have been determined as in this illustrative example, score determination system 60 will determine a composite score for each of the possible combinations of OSGi bundles by summing the corresponding products.

Thus, for the three combinations, the products would sum to yield the following composite scores:

$$\text{Combination 1: } 0 + 3.0 + 0 + 0 + 0 + 0 + 0 + 250 + 0.64 = 253.64$$

$$\text{Combination 2: } 0 + 2.0 + 0 + 0 + 0 + 0 + 0 + 350 + 0.72 = 352.72$$

$$\text{Combination 3: } 0 + 4.0 + 0 + 0 + 0 + 0 + 0 + 400 + 1.0 = 405.00$$

[0036] In general, the combination that has the lowest composite score (i.e., that which would consume the lowest amount of computer resources), is the most favorable combination (or least unfavorable) to load on client device 14A. Accordingly, based on the above analysis, bundle system 62 will select combination “1” as the most appropriate combination of OSGi bundles for loading on client device 14A. As indicated above, any of the potential combinations (e.g., 1, 2 or 3) could include OSGi bundles with or without OSGi bundle 18. Regardless, this resolution process will be implemented recursively

until all prerequisites are resolved, or it is determined that the prerequisites cannot be resolved within the resource limitations of client device 14A. For example, it could be the case the most appropriate combination of OSGi bundles itself has prerequisites that must be examined.

[0037] It should be noted that if the combinations of OSGi bundles determined to be the most appropriate to load based on the configuration of resolution engine 16 has missing prerequisites that cannot be satisfied by the bundles to be loaded, the client, or additional bundles available for distribution, then the next most appropriate combination of bundles will be tried (i.e., the possible combinations of bundles need to satisfy a set of missing prerequisites needed by the bundle(s) to be distributed yet missing from client device 14A, are tried in sorted order, based on the aforementioned calculations which mathematically describe the “undesirability” of a combination of bundles). Additionally, since all information received from client device 14A is cached on server 12, the same questions need not be asked again. For example, if it is already determined that client device 14A lacks service “A,” client device 14A will not be polled again with respect to service “A.”

[0038] In any event, when all prerequisites are completely resolved, bundle system 62 will load the final combination of OSGi bundles on client device 14A. To this extent, bundle system 62 could include an instruction passing system (not shown) that will generate and pass an instruction to client device 14A regarding the order in which the final combination of OSGi bundles should be loaded. If, however, all prerequisites could not be resolved, none of the OSGi bundles 18 or 20 will be loaded on client device 14A.

[0039] Referring now to Fig. 3, a method flow diagram 100 according to the present invention is shown. As depicted, first step S1 is to identify OSGi bundle to be loaded on a client device. Second step S2 is to determine whether the OSGi bundle has any prerequisites. If not, the OSGi bundle will be loaded on the client device in step S8. However, if there are prerequisites identified in step S2, the client device will be polled in step S3. In step S4, information from the response by the client device will be cached at the server. In step S5, it will be determined whether any of the prerequisites are missing from the client device. If not, the OSGi bundle will be loaded on the client device in step S8. If, however, prerequisites were missing from the client device, it will be determined whether there are any combinations of OSGi bundles registered with the sever that will provide the missing prerequisites in step S6. If not, the process will end without the OSGi bundle being loaded on the client device in step S9. If however, there were combinations of OSGi bundles that could resolve the missing prerequisites, a most appropriate combination of OSGi bundles will be determined in step S7. This is typically determined from the configuration file for the client device following the mathematical computations set forth in the detailed example above. Once the most appropriate combination of OSGi bundles is determined, the process will be repeated recursively from step S2 for that combination of OSGi bundles. Once all prerequisites have been resolved, the OSGi bundle and any needed additional combination(s) of OSGi bundles will be loaded on the client device in step S8.

[0040] It should be understood that the present invention can be realized in hardware, software, or a combination of hardware and software. Any kind of computer system(s) -

or other apparatus adapted for carrying out the methods described herein - is suited. A typical combination of hardware and software could be a general purpose computer system with a computer program that, when loaded and executed, carries out the respective methods described herein. Alternatively, a specific use computer, containing specialized hardware for carrying out one or more of the functional tasks of the invention, could be utilized. The present invention can also be embedded in a computer program product, which comprises all the respective features enabling the implementation of the methods described herein, and which - when loaded in a computer system - is able to carry out these methods. Computer program, software program, program, or software, in the present context mean any expression, in any language, code or notation, of a set of instructions intended to cause a system having an information processing capability to perform a particular function either directly or after either or both of the following: (a) conversion to another language, code or notation; and/or (b) reproduction in a different material form.

[0041] The foregoing description of the preferred embodiments of this invention has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed, and obviously, many modifications and variations are possible. Such modifications and variations that may be apparent to a person skilled in the art are intended to be included within the scope of this invention as defined by the accompanying claims. For example, the illustrative representation of resolution engine 16 shown in Fig. 2 is not intended to be limiting. That is, the functions

of the present invention described herein could be represented by a different configuration of systems.